


```
RRRRRRRR      MM      MM      SSSSSSSS      000000      RRRRRRRR      NN      NN      DDDDDDDD      WW      WW      NN      NN
RRRRRRRR      MM      MM      SSSSSSSS      000000      RRRRRRRR      NN      NN      DDDDDDDD      WW      WW      NN      NN
RR      RR      MMMM      MMMM      SS      00      00      RR      RR      NN      NN      DD      DD      WW      WW      NN      NN
RR      RR      MMMM      MMMM      SS      00      00      RR      RR      NN      NN      DD      DD      WW      WW      NN      NN
RR      RR      MM      MM      SS      00      0000      RR      RR      NNNN      NN      DD      DD      WW      WW      NNNN      NN
RR      RR      MM      MM      SS      00      0000      RR      RR      NNNN      NN      DD      DD      WW      WW      NNNN      NN
RRRRRRRR      MM      MM      SSSSSS      00      00      00      RRRRRRRR      NN      NN      NN      DD      DD      WW      WW      NN      NN
RRRRRRRR      MM      MM      SSSSSS      00      00      00      RRRRRRRR      NN      NN      NN      DD      DD      WW      WW      NN      NN
RR      RR      MM      MM      SS      0000      00      RR      RR      NN      NN      NN      DD      DD      WW      WW      NN      NN
RR      RR      MM      MM      SS      0000      00      RR      RR      NN      NN      NN      DD      DD      WW      WW      NN      NN
RR      RR      MM      MM      SS      00      00      RR      RR      NN      NN      NN      DD      DD      WWWW      WWWW      NN      NN
RR      RR      MM      MM      SSSSSSSS      000000      RR      RR      NN      NN      NN      DDDDDDDD      WW      WW      NN      NN
RR      RR      MM      MM      SSSSSSSS      000000      RR      RR      NN      NN      NN      DDDDDDDD      WW      WW      NN      NN
                                                                ....
                                                                ....
                                                                ....
                                                                ....
```

```
LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS
```

(2) 70
(3) 102

DECLARATIONS
RMSRMSRUNDOWN - RMS I/O RUN DOWN

```
0000 1          $BEGIN RMSORNDWN,001,RMSRMS,<RMS IO RUN DOWN>
0000 2
0000 3
0000 4 *****
0000 5 *
0000 6 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 *  ALL RIGHTS RESERVED.
0000 9 *
0000 10 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 *  TRANSFERRED.
0000 16 *
0000 17 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 *  CORPORATION.
0000 20 *
0000 21 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *****
0000 25
0000 26
0000 27 ++
0000 28 Facility: rms32
0000 29
0000 30 Abstract: this module insures all rms i/o activity is complete,
0000 31          closes all files, and resets the ifab and irab tables.
0000 32
0000 33 Environment:
0000 34          star processor running starlet exec.
0000 35
0000 36 Author: l f laverdure,          creation date: 5-5-77
0000 37
0000 38 Modified By:
0000 39
0000 40 V04-001 RAS0332          Ron Schaefer          14-Sep-1984
0000 41          ALWAYS re-enable ASTS when stalling inside rundown
0000 42          as they could get disabled by the previous exec mode
0000 43          thread of RMS that will never continue and re-enable them.
0000 44
0000 45 V03-005 DGB0040          Donald G. Blair          02-May-1984
0000 46          If the PIO$V_INHAST bit is set when we start an
0000 47          RMS operation, we conclude that the caller must be
0000 48          at exec AST level or higher and would break RMS
0000 49          synchronization rules if he were allowed to continue.
0000 50          Return error. This fix also includes a change from
0000 51          Jim Johnson to clear the FID correctly in GETDVIFID.
0000 52
0000 53 V03-004 SHZ0001          Stephen H. Zalewski          14-Sep-1983
0000 54          Move routine RMSGETDVIFID from module RMOGETDVI to here, and
0000 55          rename it GETDVIFID. Module RMOGETDVI has been evaporated.
0000 56
0000 57 V03-003 JWH0107          Jeffrey W. Horn          24-Sep-1982
```

RMSORNDWN
V04-001

RMS IO RUN DOWN

H 13

16-SEP-1984 01:29:13 VAX/VMS Macro V04-00
14-SEP-1984 22:32:57 [RMS.SRC]RMSORNDWN.MAR;2

Page 2
(1)

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :--
0000 68

Add call to RMSRU UNLOCK to release locks
held for the duration of a recovery unit.

V03-002 KBT0316 Keith B. Thompson 8-Sep-1982
Remove all S0 sharing code

V03-001 KBT0191 Keith B. Thompson 23-Aug-1982
Reorganize psects and rename entry points to single '\$'

```
0000 70      .SBTTL  DECLARATIONS
0000 71
0000 72  ::
0000 73  :: Include Files:
0000 74  ::
0000 75  ::
0000 76  ::
0000 77  :: Macros:
0000 78  ::
0000 79
0000 80      $DEVDEF
0000 81      $FABDEF
0000 82      $FIBDEF
0000 83      $FWADEF
0000 84      $IFBDEF
0000 85      $IRBDEF
0000 86      $IMPDEF
0000 87      $NWADEF
0000 88      $PIODEF
0000 89      $PSLDEF
0000 90      $RLBDEF
0000 91      $RMSDEF
0000 92
0000 93  ::
0000 94  :: Equated Symbols:
0000 95  ::
0000 96
0000 97  ::
0000 98  :: Own Storage:
0000 99  ::
0000 100
```

```
0000 102 .SBTTL RMS$RMSRUNDOWN - RMS I/O RUN DOWN
0000 103
0000 104 :++
0000 105 : RMS$RMSRUNDOWN - RMS I/O run down
0000 106
0000 107 : this routine first determines the type of rundown desired, based
0000 108 : upon the second argument. if the type is 'abort rms i/o', a branch
0000 109 : is made to rm$last_chance, otherwise the routine checks that all ifabs and irabs
0000 110 : are inactive. if any found active this routine awaits their completion after
0000 111 : first performing a $cancel i/o if not a file-oriented device.
0000 112 : when all i/o activity for the file is complete, $close is
0000 113 : performed for the file. if the close failed for an output file
0000 114 : on a files-oriented device, an error is returned to the caller
0000 115 : who should note the error and recall this routine to run down
0000 116 : further files. if all files are successfully run down the
0000 117 : image ifab & irab tables are reset and return is made to the
0000 118 : caller with a success code.
0000 119
0000 120 : files are run down in this order:
0000 121
0000 122 : 1. indirect process permanent files
0000 123 : ('error' should be first)
0000 124 : 2. image files
0000 125 : 3. (only if caller's mode is not user and arg2=1)
0000 126 : process permanent files
0000 127
0000 128 : Calling sequence:
0000 129
0000 130 : calls #2, sys$rmsrundwn
0000 131
0000 132 : Input Parameters:
0000 133
0000 134 : ap users argument list (2 arguments)
0000 135
0000 136 : arg1 descriptor for 22-character buffer
0000 137 : to receive information about
0000 138 : unsuccessfully closed output file
0000 139 : (device id and file id)
0000 140 : arg2 rundown type, as follows:
0000 141
0000 142 : 0 - run down of image and indirect i/o for process permanent files
0000 143 : 1 - run down of image and process permanent files
0000 144 : (caller's mode must be other than user)
0000 145 : 2 - abort rms i/o (caller's mode must be exec or kernel)
0000 146
0000 147 : all others are reserved, but currently behave as type 0
0000 148
0000 149 : Implicit Inputs:
0000 150
0000 151 : caller's mode.
0000 152
0000 153 : Output Parameters:
0000 154
0000 155 : r0 status code
0000 156 : r1 destroyed
0000 157
0000 158 : Implicit Outputs:
```

```
0000 159 :  
0000 160 : information describing an output file unsuccessfully closed is  
0000 161 : stored in the caller-provided buffer in exactly the same  
0000 162 : format as the dvi, did, and fid fields of the nam block.  
0000 163 :  
0000 164 : Completion Codes:  
0000 165 :  
0000 166 :     standard rms, in particular:  
0000 167 :  
0000 168 :     rms$_suc - all files closed  
0000 169 :     rms$_ccf - an output file could not be closed  
0000 170 :                  successfully. caller-provided buffer  
0000 171 :                  has information identifying the file  
0000 172 :     rms$_ial - same as rms$_ccf except could not  
0000 173 :                  access caller's buffer to store file  
0000 174 :                  id information.  
0000 175 :  
0000 176 : Side Effects:  
0000 177 :  
0000 178 :     runs synchronously in exec mode inhibiting  
0000 179 :     and enabling asts as required.  
0000 180 :  
0000 181 :--  
0000 182 :
```

```
0000 184 $ENTRY RMS$RMSRUNDOWN
0000 185 $TSTPT RUNDWN
5B DC 0006 186 MOVPSL R11
16 EF 0008 187 EXTZV #PSLSV-PRVMD,-
57 5B 02 000A 188 #PSLSS-PRVMD,R11,R7 ; save caller's mode
02 08 AC D1 000D 189 CMPL 8(AP),#2 ; abort rms i/o?
42 13 0011 190 BEQL RMSABORT ; branch if yes
00 E2 0013 191 BBSS #PIO$V-INHAST,- ; br if RMS already in progress
40 00000000'9F 0015 192 @#PIO$GW-STATUS,ERRBUSY
001B 193
001B 194 :
001B 195 : start by releasing locks held for the durrantion of a recovery unit,
001B 196 : if any.
001B 197 :
001B 198
FFE2' 30 001B 199 BSBW RMSRU_UNLOCK
001E 200
001E 201 :
001E 202 : next run down indirect i/o on process-permanent files
001E 203 :
001E 204
5B 00000000'9F DE 001E 205 MOVAL @#PIO$GW-PIOIMPA,R11 ; get pio impure area address
58 01 D0 0025 206 ASSUME IMP$W-RMSSTATUS EQ 0
006B 30 0025 207 MOVL #1,R8 ; indicate indirect run down
0028 208 BSBW RUNDWN ; do the run down
002B 209 ; (note: clears r8)
002B 210 :
002B 211 : now run down the image
002B 212 :
002B 213
5B 0000'CB DE 002B 214 MOVAL W^PIO$GW-IIIOIMPA-PIO$GW-PIOIMPA(R11),R11
0030 215
0030 216 :
0030 217 : point to image impure area
0030 218 :
0030 219
0063 30 0030 220 BSBW RUNDWN ; do the run down
18 BB D4 0033 221 CLRL @IMP$L-IFABTBL(R11) ; reset ifab table link
1C BB D4 0036 222 CLRL @IMP$L-IRABTBL(R11) ; reset irab table link
0039 223
0039 224 :
0039 225 : point to process
0039 226 : i/o impure area again
0039 227 :
0039 228
5B 0000'CB DE 0039 229 MOVAL W^PIO$GW-PIOIMPA-PIO$GW-IIIOIMPA(R11),R11
003E 230 :
003E 231 : At this point there used to be code to return any whole pages
003E 232 : on the FMLH free space list back to the process i/o free page
003E 233 : list. The space on the FMLH list is currently (v 2) used only
003E 234 : for ASB allocation on IFAB operations and will bugcheck if space
003E 235 : is not found. The behavior is now that a page will be added to
003E 236 : the FMLH list the first time a process stalls on an IFAB operation
003E 237 : and will remain there for the life of the process.
003E 238 :
003E 239 :
003E 240 :
```

```
01 08 AC 91 003E 241 ; now run down direct i/o on process-permanent files if desired
      07 12 003E 242 ;
      03 57 91 003E 243 ;
      02 13 0042 244 CMPB 8(AP),#1 ; ppf rundown?
      4B 10 0042 245 BNEQ XITSUC ; branch if not
      0044 246 CMPB R7,#PSL$C_USER ; caller sufficiently privileged?
      0047 247 BEQL XITSUC ; branch if not
      0049 248 60$: BSBB RUNDWN ; do the run down
      004B 249 XITSUC: RMSSUC
      004E 250 EXIT:
      11 10 004E 251 SSB #16, R0 ; stamp 'rms' on status code
      04 0052 252 BSBB ENBAST ; enable asts
      0054 253 RET ; back to caller
      0055 254
      0055 255 ;
      0055 256 ; branch to rm$last_chance to do async process deletion rms i/o abort
      0055 257 ;
      0055 258
      00000000'EF 17 0055 259 RMSABORT:
      0055 260 JMP RMSLAST_CHANCE
```

```
005B 262 :  
005B 263 : If the PIO$V_INHAST bit is already set, we  
005B 264 : conclude that the caller must be at exec ast level or higher  
005B 265 : (otherwise, he could not have kicked off an RMS operation  
005B 266 : while RMS was already in progress) and would break RMS  
005B 267 : synchronization rules if allowed to continue. Return RMS$_BUSY  
005B 268 : status when this happens.  
005B 269 :  
005B 270 :  
005B 271 ERRBUSY:  
005B 272     RMSERR  BUSY  
0060 273     SSB    #16,R0  
04  0064 274     RET  
0065 275 :  
0065 276 : enable rms ast's, reenabling exec ast's in all cases.  
0065 277 :  
0065 278 :  
0065 279 :  
0065 280 ENBAST: CSB    #PIO$V_INHAST, @#PIO$GW_STATUS  
006D 281 :  
006D 282 : clear ast inhibit and enable asts  
006D 283 :  
006D 284 :  
006D 285 :  
006D 286     $SETAST_S    #1                ; enable exec mode asts  
05  0076 287     RSB  
0077 288 :  
0077 289 : inhibit rms asts  
0077 290 :  
0077 291 :  
0077 292 :  
00000000'9F 01  A8 0077 293 INHAST: BISW2    #1@PIO$V_INHAST, @#PIO$GW_STATUS  
05  007E 294     RSB  
007F 295 :  
007F 296 :  
007F 297 : wait for rms operation completion  
007F 298 :  
007F 299 :  
DB  10 007F 300 WAIT:  $CLREF_S    #IMP$C_IOREFN    ; clear rms event flag  
0088 301     BSBB    ENBAST    ; enable asts  
008A 302     $WAITFR_S    #IMP$C_IOREFN    ; wait for flag  
E2  10 0093 303     BSBB    INHAST    ; re-inhibit asts  
05  0095 304     RSB
```

```
0096 306
0096 307 :++
0096 308
0096 309 : run down subroutine:
0096 310
0096 311 : checks ifab table for active files.
0096 312 : if any found waits for any i/o activity to finish
0096 313 : (doing a cancell i/o for non files-oriented devices)
0096 314 : and then issues a $close request.
0096 315
0096 316 : when all files run down performs a sanity check by seeing if all irab
0096 317 : table entries are also zero.
0096 318
0096 319 : inputs:
0096 320 : r11 - impure area addr
0096 321 : r8 - bit 0 set if indirect ppf run down
0096 322 : ap - caller's arg list
0096 323 : r7 - caller's mode
0096 324
0096 325 : outputs:
0096 326 : returns only if noerror encountered.
0096 327 : imp$V_ppfindrd cleared
0096 328 : r0 - r6, r9, r10 destroyed
0096 329 :--
0096 330
0096 331 RUNDWN:
0096 332 SSB #IMP$V_IORUNDOWN,(R11) : set i/o rundown in progress flag
009A 333 : to sync with ast-driven rms
009A 334 : operations
009A 335 MOVL IMP$L_IFABTBL(R11),R5 : get ifab table addr
009E 336 CLRL R6 : build ifi value here
55 18 AB D0 00A0 337 NXTSEG: MOVL (R5)+,R2 : save addr next table seg in r2
52 85 D0 00A3 338 MOVZWL IMP$W_ENTPERSEG(R11),R4 : get # entries/seg
54 20 AB 3C 00A7 339 NXTENT: TSTW IMP$W_NUM_IFABS(R11) : any ifabs active?
22 AB B5 00AA 340 BEQL CHKIRB : branch if none
OF 13 00AC 341 INCL R6 : bump ifi
56 D6 00AE 342 MOVL (R5)+,R10 : get ifab addr
5A 85 D0 00B1 343 BNEQ RDIFAB : branch if one
2F 12 00B3 344 NXTSOB: SOBGTR R4,NXTENT : keep scanning segment
F1 54 F5 00B6 345
00B6 346 :
00B6 347 : no more ifabs this segment, try next
00B6 348 :
00B6 349
55 52 D0 00B6 350 MOVL R2,R5 : get next segment addr
E5 12 00B9 351 BNEQ NXTSEG : branch if one
```

```
00BB 353
00BB 354
00BB 355 : all ifabs have been run down now.
00BB 356
00BB 357 : unless this is indirect run down of ppf's,
00BB 358 : check that all irabs are also gone.
00BB 359
00BB 360
00BB 361 CHKIRB:
00BB 362 .IF NE $RMS$TEST&$RMS$_TBUGCHK
17 58 00 E4 00BB 363 BBSC #0,R8,30$ : branch if indirect run down
55 1C AB D0 00BF 364 MOVL IMP$L_IRABTBL(R11),R5 : get irab table addr
54 52 85 D0 00C3 365 10$: MOVL (R5)+,R2 : save addr next table seg.
20 AB 3C 00C6 366 MOVZWL IMP$W_ENTPERSEG(R11),R4 : get # entries/seg.
85 D5 00CA 367 20$: TSTL (R5)+ : entry zero?
0D 12 00CC 368 BNEQ ERRBUG : branch if not
F9 54 F5 00CE 369 SOBGTR R4,20$ : branch if more entries
55 52 D0 00D1 370 MOVL R2,R5 : get next seg addr
ED 12 00D4 371 BNEQ 10$ : branch if one
00D6 372 30$: CSB #IMP$V_IORUNDOWN,(R11) : turn off rundown in progress flag
00DA 373 .ENDC
05 0CDA 374 RSB : all o.k.
00D8 375
00D8 376
00D8 377 : close failed to zero ifab or irab table entry
00D8 378
00D8 379 ERRBUG: RMSTBUG FTL$_IORNDN
```

```
00E2 381
00E2 382
00E2 383 : found an ifab. check for active and if so allow operation to finish
00E2 384 :
00E2 385
00E2 386 ASSUME IMP$W_RMSSTATUS EQ 0
00E2 387 RDIFAB:
26 6A 2A 58 E8 00E2 388 BLBS R8,RDNET : branch if indirect ppf
04 6A 20 E1 00E5 389 10$: BBC #IFB$V_BUSY,(R10),RDNET : if not busy then check NETWORK
04 6A 0D E0 00E9 390 BBS #DEV$V_NET,(R10),20$ : do cancel if busy & network operation
30 6A 3A E1 00ED 391 BBC #IFB$V_RMS_STALL,- : if this RMS thread is not currently
00EF 392 (R10),RDIRAB : stalled then skip the cancel and wait
00F1 393
00F1 394
00F1 395 : allow function to finish
00F1 396 : \note: this code should be modified to
00F1 397 : properly run down read-ahead and write-behind
00F1 398 : operations to unit record devices.\
00F1 399
00F1 400
00F1 401 20$: $CANCEL_S IFB$W_CHNL(R10) : cancel i/o (e.g. magtape create)
00FC 402 BSBB WAIT : wait for an operation to finish
FC A5 D5 00FE 403 TSTL -4(R5) : ifab disappear? (close)
08 6A 20 E1 0101 404 BEQL NXTSOB : branch if yes
E6 6A 0D E0 0103 405 BBC #IFB$V_BUSY,(R10),RDNET : run down NETWORK if no longer busy
04 6A 3A E0 0107 406 BBS #DEV$V_NET,(R10),20$ : but do cancel & wait again if busy &
E2 6A 010B 407 BBS #IFB$V_RMS_STALL,- : network operation or busy and the RMS
010D 408 (R10),20$ : thread is still stalled
010F 409
010F 410
010F 411 : if the current operation is a network operation, and a special recieve QIO
010F 412 : has been posted but NOT recieved, a $CANCEL must always be done to flush
010F 413 : this QIO. In file transfer mode it will be possible that a recieve has been
010F 414 : posted but no transfer operation is underway. therefore neither the IFAB nor
010F 415 : the IRAB will be busy. if a $CANCEL isn't explicitly issued, when the $CLOSE
010F 416 : is performed, the NETDRIVER will be unable to disconnect the logical link
010F 417 : (because of the outstanding recieve), and the process will hang.
010F 418
010F 419
0E 6A 0D E1 010F 420 RDNET: BBC #DEV$V_NET,(R10),RDIRAB : go run down IRABs if not network op
50 3C AA D0 0113 421 MOVL IFB$L_NWA_PTR(R10),R0 : obtain network work area address
08 13 0117 422 BEQL RDIRAB : skip check if not network work area
03 E1 0119 423 BBC #NWA$V_RCVQIO,- : if a special recieve QIO has not been
04 60 011B 424 (R0),RDIRAB : posted go run down the IRABs, but if
04 E1 011D 425 BBC #NWA$V_RCVAST,- : one has and it hasn't been recieved
29 60 011F 426 (R0),CANCEL : then go issue the cancel
0121 427
0121 428 : run down irabs
0121 429 :
0121 430 :
0121 431
0121 432 RDIRAB:
59 5A D0 0121 433 MOVL R10,R9 : copy ifab addr
59 1C A9 D0 0124 434 10$: MOVL IRB$L_IRAB_LNK(R9),R9 : get next irab
03 30 13 0128 435 BEQL QUIET : branch if none
03 58 E8 012A 436 BLBS R8,12$ : don't release locks if indirect PPF
012D 437 : rundown
```

```
FED0' 30 012D 438 BSBW RMSUNLOCKALL ; kill all record locks, including
                                0130 439 ; outstanding waits.
FO 69 20 E1 0130 440 12$: BBC #IRBSV_BUSY,(R9),10$ ; branch if idle
04 6A 0D E0 0134 441 BBS #DEVSV_NET,(R10),15$ ; do cancel if busy & network operation
      3A E1 0138 442 BBC #IRBSV_RMS_STALL,- ; if this RMS thread is not currently
      E8 69 013A 443 (R9),10$ ; stalled then skip the cancel and wait
                                013C 444
                                013C 445 15$: BLBS (R11), 20$ ; branch if image i/o segment
03 69 22 E0 013F 446 BBS #IRBSV_PPF_IMAGE,(R9),20$
                                0143 447
                                0143 448 ;
                                0143 449 ; branch if indirect i/o
                                0143 450 ;
                                0143 451
DE 58 E9 0143 452 BLBC R8,10$ ; branch if only indirect ppfs
      1C E0 0146 453 ; to be run down
      6A 0146 454 20$: BBS #DEVSV_RND,-
      0B 0148 455 IFBSL PRIM_DEV(R10),- ; no need to do a cancel if this is
      0149 456 NOCANCEL ; a disk operation, just go wait
      014A 457
      014A 458 CANCEL: $CANCEL_S IFBSW_CHNL(R10) ; cancel i/o
      0155 459
      0155 460 NOCANCEL:
FF27 30 0155 461 BSBW WAIT ; wait for all ASTs to be delivered
      C7 11 0158 462 BRB RDIRAB ; start from top of irab
      015A 463 ; chain again (could
      015A 464 ; have been disconnect)
```

```
015A 466
015A 467 : all activity ceased for this file.
015A 468 : force a close by constructing a fab and calling close.
015A 469 :
015A 470 :
015A 471 :
78 AA D5 015A 472 QUIET: TSTL IFB$S_SFSB_PTR(R10) ; is it a shared file?
OC 12 015D 473 BNEQ 5$ ; yes, go close it
35 6A 30 E1 015F 474 BBC #IFB$V_WRTACC,(R10),NOERR ; branch if not write access
31 6A 03 E1 0163 475 BBC #DEV$V_DIR,IFB$S_PRIM_DEV(R10),NOERR
20 6A 25 E1 0167 476 BBC #IFB$V_ACCESSED,(R10),NOERR ; branch if file not accessed
50 04 AC D0 016B 477 5$: MOVL 4(AP),R0 ; get descriptor addr
016F 478 IFNORD #8,(R0),NOERR1,R7
1C 60 B1 0175 479 CMPW (R0),#28 ; at least 22 bytes long?
17 1F 0178 480 BLSSU NOERR1
53 04 A0 D0 017A 481 MOVL 4(R0),R3 ; get buffer address
59 5A D0 017E 482 MOVL R10,R0 ; ifab to right register
0181 483 IFNOWRT #22,(R3),NOERR1,R7 ; branch if buffer not writable
0069 30 0187 484 BSBW GETDVI_FID ; go fill buffer with dvi and fid
0A 11 018A 485 RMSERR CCF,R3 ; get set for close failure
018F 486 BRB CLOSE
0191 487
0191 488 NOERR1: RMSERR IAL,R3 ; if close failure, return ial
03 11 0196 489 BRB CLOSE
0198 490
0198 491 NOERR: RMSSUC SUC,R3 ; can't fail
019B 492
5E B0 AE DE 019B 493 CLOSE: MOVAL -FAB$C_BLN(SP),SP ; create fab on stack
S003 8F B0 019F 494 MOVW #FAB$C_BID+<FAB$C_BLN @8>,- ;
6E 01A3 495 (SP) ; fab block id and length
02 AE 56 B0 01A4 496 MOVW R6,FAB$W_IFI(SP) ; ifi
0B 6B E8 01A8 497 BLBS (R11),10$ ; branch if iio seg
04 5B E9 01AB 498 SSB #15+<FAB$W_IFI*8>,(SP) ; set pio flag
01AF 499 BLBC R8,10$ ; branch if direct access
01B2 500 SSB #FAB$V_PPF_IND+<FAB$W_IFI*8>,- ;
01B2 501 (SP) ; else make indirect ifi
01B6 502 10$: PUSHR #^M<R2,R3,R4,R5> ; save regs
00 6E 00 2C 01B8 503 MOVCS #0,(SP),#0,- ;
14 AE 004C 8F 01BC 504 #FAB$C_BLN-4,4+<4*4>(SP); zero remainder of fab
3C BA 01C1 505 POPR #^M<R2,R3,R4,R5> ; restore r5
FE9F 30 01C3 506 BSBW ENBAST ;
SE DD 01C6 507 PUSHL SP ; addr of fab
00000000'9F 01 01C8 508 CALLS #1,@#SYS$CLOSE ; close it
FEAS 30 01CF 509 BSBW INHAST ;
5E 00000050 8F C0 01D2 510 ADDL #FAB$C_BLN,SP ; 'pop' fab
05 58 E8 01D9 511 BLBS R8,15$ ; omit check if indirect ppf
FC A5 D5 01DC 512 TSTL -4(R5) ; did ifab go away?
OF 12 01DF 513 BNEQ ERRBUG_BR ; branch if not
03 50 E9 01E1 514 15$: BLBC R0,30$ ; branch on error
FECC 31 01E4 515 20$: BRW NXISOB ; get next ifab
50 53 D0 01E7 516 30$: MOVL R3,R0 ; get saved error code
F7 50 E8 01EA 517 BLBS R0,20$ ; no problem if not
FESE 31 01ED 518 BRW EXIT ; write-accessed file
FEEB 31 01F0 519 ERRBUG_BR: ; return error to caller
01F0 520 BRW ERRBUG ; extended branch
01F3 521
01F3 522
```

```
01F3 524 :++
01F3 525 : GETDEVIFID -- Get Device ID and File ID.
01F3 526 :
01F3 527 :   This routine returns the counted device name string,
01F3 528 :   as well as the file id for the file open on the channel.
01F3 529 :
01F3 530 : Calling Sequence:
01F3 531 :
01F3 532 :   BSBW   GETDVIFID
01F3 533 :
01F3 534 : Input Parameters:
01F3 535 :
01F3 536 :   R9      IFAB address
01F3 537 :   R3      address of 22-byte buffer to return device name string
01F3 538 :   IFBSW_CHNL channel #
01F3 539 :
01F3 540 : Implicit Inputs:
01F3 541 :
01F3 542 :   none
01F3 543 :
01F3 544 : Output Parameters:
01F3 545 :
01F3 546 :   R0,R1,R3   destroyed
01F3 547 :
01F3 548 : Implicit Outputs:
01F3 549 :
01F3 550 :   The counted ascii string for the device name is moved
01F3 551 :   to the buffer provided, followed by the file id starting 16 bytes
01F3 552 :   from the start of the buffer.
01F3 553 :
01F3 554 : --
01F3 555 :
```

```
0434 8F BB 01F3 557 GETDVIFID:
      53 DD 01F3 558          PUSHR #^M<R2,R4,R5,R10>      ; Save regs.
      38 A9 D0 01F7 559          PUSHL R3                  ; Save R3.
83 5A 0190 CA 90 01F9 560          MOVL IFBSL_FWA_PTR(R9),R10 ; Get FWA into R10.
      0190 CA 90 01FD 561          MOVB FWASQ_SHRFIL(R10),(R3)+ ; Move size of buffer id into first byte of
      0194 DA 28 0202 562          MOVCS FWASQ_SHRFIL(R10),- ; Move device id name into buffer
      63      0206 563          @FWASQ_SHRFIL+4(R10),-
      53 BED0 0209 564          (R3)
      020A 565          POPL R3                          ; Restore R3.
      020D 566
      020D 567
      020D 568
      020D 569
      020D 570
      020D 571
      020D 572
      020D 573          CLRL 16(R3)                        ; Clear FID field in buffer.
      14 A3 D4 0210 574          CLRW 20(R3)
07 69 05 B4 0213 575          BBS #DEV$V_SQD,(R9),10$      ; branch if magtape (no FCB)
      06 28 0217 576          MOVCS #6,-                  ; Move FID to buffer.
      01F8 CA 0219 577          FWAST_FIBBUF+FIBSW_FID(R10),-
      10 A3 021C 578          16(R3)
0434 8F BA 021E 579 10$: POPR #^M<R2,R4,R5,R10>      ; Restore regs.
      05 0222 580          RSB
      0223 581
      0223 582          .END
```

RMSORNDWN
Symbol table

RMS IO RUN DOWN

I 14

16-SEP-1984 01:29:13 VAX/VMS Macro V04-00
14-SEP-1984 22:32:57 [RMS.SRC]RMSORNDWN.MAR;2

Page 16
(15)

```

$$PSECT_EP      = 00000000
$$RMSTEST       = 0000001A
$$RMS_PBUGCHK   = 00000010
$$RMS_TBUGCHK   = 00000008
$$RMS_UMODE     = 00000004
CANCEC         = 0000014A R    01
CHKIRB         = 00000088 R    01
CLOSE          = 0000019B R    01
DEVSV_DIR      = 00000003
DEVSV_NET      = 0000000D
DEVSV_RND      = 0000001C
DEVSV_SQD      = 00000005
ENBAST         = 00000065 R    01
ERRBUG         = 000000DB R    01
ERRBUG_BR      = 000001F0 R    01
ERRBUSY        = 0000005B R    01
EXIT           = 0000004E R    01
FABSC_BID      = 00000003
FABSC_BLN      = 00000050
FABSV_PPF_IND  = 0000000E
FABSW_IFI      = 00000002
FIBSW_FID      = 00000004
FTLS_IORNDN    = 00000000
FWASQ_SHRFIL   = 00000190
FWAST_FIBBUF   = 000001F4
GETDVIFID      = 000001F3 R    01
IFBSL_FWA_PTR  = 00000038
IFBSL_NWA_PTR  = 0000003C
IFBSL_PRIM_DEV = 00000000
IFBSL_SFSB_PTR = 00000078
IFBSV_ACCESSED = 00000025
IFBSV_BUSY     = 00000020
IFBSV_RMS_STALL = 0000003A
IFBSV_WRTACC   = 00000030
IFBSW_CHNL     = 00000020
IMPSC_IOREFN   = 0000001E
IMPSL_IFABTBL  = 00000018
IMPSL_IRABTBL  = 0000001C
IMPSV_IORUNDOWN = 00000004
IMPSW_ENTPERSEG = 00000020
IMPSW_NUM_IFABS = 00000022
IMPSW_RMSSTATUS = 00000000
INHAST         = 00000077 R    01
IRBSL_IRAB_LNK = 0000001C
IRBSV_BUSY     = 00000020
IRBSV_PPF_IMAGE = 00000022
IRBSV_RMS_STALL = 0000003A
NOCANCEL       = 00000155 R    01
NOERR          = 00000198 R    01
NOERR1         = 00000191 R    01
NWA$B_ALLXABCNT = 0000011C
NWA$B_DAP_RAC  = 000000C9
NWA$B_FILESYS  = 000000C5
NWA$B_KEYXABCNT = 0000011D
NWA$B_NETSTRSIZ = 0000016F
NWA$B_NODBUFSIZ = 00000168
NWA$B_ORG      = 000000C6

```

```

NWA$B_OSTYPE   = 000000C4
NWA$B_RFM      = 000000C7
NWA$B_RMS_RAC  = 000000C8
NWA$C_BLN      = 00000800
NWA$K_BLN      = 00000800
NWA$L_ALLXABADR = 00000100
NWA$L_DATXABADR = 00000104
NWA$L_DEV      = 000000C0
NWA$L_FHCXABADR = 00000108
NWA$L_KEYXABADR = 0000010C
NWA$L_MSG_MASK = 000000D4
NWA$L_PROXABADR = 00000110
NWA$L_RDTXABADR = 00000114
NWA$L_SAVE_FLGS = 00000128
NWA$L_SUMXABADR = 00000118
NWA$L_THREAD   = 000000FC
NWA$L_XLTATTR  = 00000238
NWA$L_XLTBUFLG = 0000022C
NWA$L_XLTCNT   = 00000228
NWA$L_XLTMAXINDX = 00000234
NWA$L_XLTSIZ   = 00000230
NWA$Q_ACS      = 00000244
NWA$Q_BIGBUF   = 00000170
NWA$Q_BLD      = 000000F0
NWA$Q_FLG      = 00000000
NWA$Q_INODE     = 0000025C
NWA$Q_IOSB     = 000000D8
NWA$Q_LNODE     = 00000160
NWA$Q_LOGNAME   = 0000023C
NWA$Q_NCB       = 00000264
NWA$Q_RCV       = 000000E0
NWA$Q_SAVE_DESC = 00000120
NWA$Q_XLTBUF1  = 0000024C
NWA$Q_XLTBUF2  = 00000254
NWA$Q_XMT       = 000000E8
NWA$T_ACSBUF    = 0000026C
NWA$T_AUXBUF    = 000005E0
NWA$T_DAP       = 00000000
NWA$T_INODEBUF  = 000004AC
NWA$T_ITM_ATTR  = 00000200
NWA$T_ITM_END   = 00000224
NWA$T_ITM_LST   = 00000200
NWA$T_ITM_MAXINDX = 00000218
NWA$T_ITM_STRING = 0000020C
NWA$T_NCBBUF    = 0000052C
NWA$T_NODEBUF   = 00000169
NWA$T_RCVBUF    = 000001A0
NWA$T_SCAN      = 00000100
NWA$T_TEMP      = 00000120
NWA$T_XLTBUF1   = 000002AC
NWA$T_XLTBUF2   = 000003AC
NWA$T_XMTBUF    = 000003C0
NWA$V_RCVAST    = 00000004
NWA$V_RCVQIO    = 00000003
NWA$W_BUILD     = 000000D2
NWA$W_DAPBUFSIZ = 000000CA
NWA$W_DIR_OFF   = 000000CC

```

RMSORNDWN
Symbol table

RMS IO RUN DOWN

J 14

16-SEP-1984 01:29:13
14-SEP-1984 22:32:57

VAX/VMS Macro V04-00
[RMS.SRC]RMSORNDWN.MAR;2

Page 17
(15)

NWASW_DISPLAY	000000D0		
NWASW_FIL_OFF	000000CE		
NWASW_JNLXABJOP	0000011E		
NXTENT	000000A7	R	01
NXTSEG	000000A0	R	01
NXTSOB	000000B3	R	01
PIO\$A_TRACE	*****	X	01
PIO\$G_IIOIMPA	*****	X	01
PIO\$G_PIOIMPA	*****	X	01
PIO\$G_STATUS	*****	X	01
PIO\$V_INHA\$T	= 00000000		
PSL\$C_USER	= 00000003		
PSL\$S_PRV\$MOD	= 00000002		
PSL\$V_PRV\$MOD	= 00000016		
QUIET	0000015A	R	01
RDIFAB	000000E2	R	01
RDIRAB	00000121	R	01
RDNET	0000010F	R	01
RMSBUG	*****	X	01
RMSLAST_CHANCE	*****	X	01
RMSRU_UNLOCK	*****	X	01
RMSUNCOCKALL	*****	X	01
RMS\$RMSRUN\$DWN	= FFFFFFFE	RG	01
RMS\$_BUSY	= 0001848C		
RMS\$_CCF	= 0001C0DC		
RMS\$_IAL	= 0001854C		
RMSABORT	00000055	R	01
RUNDWN	00000096	R	01
SY\$CANCEL	*****	GX	01
SY\$CLOSE	*****	X	01
SY\$CLREF	*****	GX	01
SY\$SETAST	*****	GX	01
SY\$WAITFR	*****	GX	01
TPT\$\$_RUNDWN	*****	X	01
WAIT	0000007F	R	01
XITSUC	0000004B	R	01

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RMSRMS	00000223 (547.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
\$ABS\$	00000800 (2048.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.08	00:00:00.84
Command processing	142	00:00:00.72	00:00:04.05
Pass 1	417	00:00:16.06	00:00:41.85
Symbol table sort	0	00:00:02.27	00:00:04.14

Pass 2	109	00:00:02.95	00:00:06.37
Symbol table output	19	00:00:00.18	00:00:00.26
Psect synopsis output	1	00:00:00.04	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	726	00:00:22.31	00:00:57.63

The working set limit was 1650 pages.

86675 bytes (170 pages) of virtual memory were used to buffer the intermediate code.

There were 90 pages of symbol table space allocated to hold 1649 non-local and 23 local symbols.

582 source lines were read in Pass 1, producing 14 object records in Pass 2.

37 pages of virtual memory were used to define 36 macros.

```

+-----+
! Macro library statistics !
+-----+

```

Macro library name	Macros defined
-\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	18
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	3
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	32

1838 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RMSORNDWN/OBJ=OBJ\$:RMSORNDWN MSRC\$:RMSORNDWN/UPDATE=(ENH\$:RMSORNDWN)+EXECML\$/LIB+LIB\$:RMS/LIB

0330

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY